## REMARKS/ARGUMENTS

The amendment to the specification corrects a clearly typographical error. No new matter has been added by the amendment to the specification.

Claims 1-22 are pending in the present application. Claims 1, 5, 13, 15, and 22 are amended. Claim 2 has been canceled, and claim 23 is new. Support for the new claim and claim amendments can be found in the claims themselves and in the Applicant's patent application on page 6, line 3 – page 11, line 9 and figures 1-3. Reconsideration of the claims is respectfully requested.

### I. 35 U.S.C. § 101

The Examiner rejected claim 22 under 35 U.S.C. § 101 as directed towards non-statutory subject matter. Applicants have amended claim 22 accordingly, thereby overcoming the rejection.

### II. 35 U.S.C. § 102, Asserted Anticipation Against Claims 1-6 and 8-22

The Examiner rejected claims 1-6 and 8-22 under 35 U.S.C. § 102 as anticipated by *Walker*, Mechanism for Converting between JAVA Classes and XML, U.S. Patent Application Publication 2004/0015840, April 19, 2001 (hereinafter referred to as "*Walker*"). The rejection is respectfully traversed.

With regard to claim 1, the Examiner states:

> Per claim 1, Walker discloses a method, in a data processing system, for code reusability and maintainability, the method comprising:
> providing a utility class in a server that defines a utility method (para [0072-0073] ". . .XmlUtil class.. .");
> responsive to receiving a request at the server for attributes for an entity from a client (parag [0091] ". . . a request is made to the API . . . name of java class . . ."), generating a method call for the utility method (para [0039] ". . . method allows a JAVA class . . ."), wherein the method call identifies the entity and a response object name (parag [0091] ". . . name of java class specified.. . ");
> generating a response object and assigning the response object name to the response object (para [0056] ". . . what class of object must be constructed . . ."); and
> returning the response object to the client (para [0097] ". . . object is returned . . .").

Office Action dated November 15, 2006, p. 3.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582,

32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

Claim 1 is representative of claim 13 and amended claim 22. *Walker* does not anticipate claim 1 because *Walker* does not disclose each and every feature of claim 1. Claim 1 as amended is as follows:

> 1. A method, in a data processing system, for code reusability and maintainability, the method comprising:
>
> providing a utility class in a server, wherein the utility class defines a utility method;
>
> responsive to receiving a request at the server for attributes for an entity from a client, generating a method call for the utility method, wherein the method call identifies the entity and a response object name;
>
> generating a response object and assigning the response object name to the response object; and
>
> returning the response object to the client.

*Walker* does not disclose all the features of claim 1. Specifically, *Walker* fails to disclose (1) the feature of receiving a request at the server for attributes for an entity from a client, (2) the feature of returning the response object to the client, and (3) the feature of "responsive to" in the feature of, "responsive to receiving a request at the server for attributes for an entity from a client, generating a method call for the utility method, wherein the method call identifies the entity and a response object name."

**II.A. *Walker* Fails to Disclose the Feature of Receiving a Request at the Server for Attributes for an Entity from a Client**

*Walker* fails to disclose the feature of receiving a request at the server for attributes for an entity from a client. The Examiner asserts otherwise, citing the following portion of *Walker*:

> [0091] 1. A request is made to the API to convert a Java class to an XML representation. The name of the Java class and the name of the top level document tag are specified.

*Walker*, paragraph 91.

Neither the cited portion nor any other portion of *Walker* discloses the feature of receiving a request at the server for attributes for an entity from a client. *Walker* discloses a mechanism for converting between JAVA classes and XML that is programmed onto a single computer system. The cited portion of *Walker* discloses a request to an application programming interface to convert a JAVA class to an XML representation. However, neither the cited portion, nor any other portion of *Walker*, discloses the source of request described therein.

On the other hand, claim 1 recites the feature of receiving a request at the server for attributes for an entity from a client. *Walker's* disclosure differs from the claimed feature of claim 1, because the claimed feature requires that a request originate from a client, while *Walker* fails to disclose any source from which the disclosed request originates. For example, the cited portion discloses only that a request "is made to the API," but fails to disclose the source of the request. Because *Walker* fails to disclose the source of the request, *Walker* cannot disclose that the request is from a client, as claimed.

Indeed, *Walker's* mechanism does not pass requests to and from a client or a server because *Walker's* invention does not rely on client/server architecture for its implementation. *Walker* discloses a JAVA/XML conversion mechanism, embodied as a single computer program, which executes on a single computer system. *Walker* does not disclose any components that approximate a client and a server. For example, the words "client" or "server" fail to appear in *Walker* at all. *Walker's* invention does not depend upon the existence of a client or a server to accomplish its objective of converting between JAVA classes and XML, and *Walker* fails to disclose otherwise. With regard to the portion of *Walker* cited by the Examiner, *Walker's* invention converts between JAVA classes and XML without regard to the source of the conversion request, and *Walker* fails to disclose any such source, let alone a client. Therefore, *Walker* fails to disclose the feature of receiving a request at the server for attributes for an entity from a client. Accordingly, *Walker* does not disclose all the features of claim 1.

### II.B. *Walker* Fails to Disclose the Feature of Returning the Response Object to the Client

*Walker* fails to disclose the feature of returning the response object to the client. The Examiner asserts otherwise, citing the following portion of *Walker*:

> [0097] 3. If the object does not implement XmlReaderWriter, the object is returned. If it does, proceed to step 4.

*Walker*, paragraph 97.

Neither the cited portion nor any other portion of *Walker* discloses the feature of returning the response object to the client. The cited portion of *Walker* describes one step of the multi-step mechanism to convert XML to a JAVA object. However, the cited portion fails to disclose the destination of the disclosed object.

On the other hand, claim 1 recites the feature of returning the response object to the client. Even assuming, *arguendo*, that the object disclosed in the cited portion is a response object, *Walker* still fails to teach the claimed feature. Specifically, *Walker's* disclosure differs from the claimed feature of claim 1, because the claimed feature requires the response object to be returned to the client, while *Walker* fails to disclose any destination for the returned object. For example, the cited portion discloses only that "the object is returned," but fails to disclose the destination of the returned object. Because *Walker* fails to disclose the destination of the returned object, *Walker* cannot disclose that the request is returned to the

client, as claimed. Also, for the same reasons stated in Section II.A., *Walker* does not pass objects between a client and a server because *Walker* does not rely on client/server architecture for its implementation. Therefore, *Walker* fails to disclose the feature of returning the response object to the client. Accordingly, *Walker* does not disclose all the features of claim 1.

**II.C. *Walker* Fails to Disclose the "Responsive To" Relationship in the Feature of Responsive to Receiving a Request at the Server for Attributes for an Entity from a Client, Generating a Method Call for the Utility Method, Wherein the Method Call Identifies the Entity and a Response Object Name**

*Walker* fails to disclose the "responsive to" relationship in the feature of responsive to receiving a request at the server for attributes for an entity from a client, generating a method call for the utility method, wherein the method call identifies the entity and a response object name. The Examiner asserts otherwise, citing paragraph 91 of *Walker*, reproduced above, as well as the following portion of *Walker*:

> [0039] The first method getFieldDescriptions is required. This method allows a JAVA class to define how it should be converted. Typically, this will add just one line of code for all sub-elements contained by the class.

*Walker*, paragraph 97.

Neither the cited portion nor any other portion of *Walker* discloses the "responsive to" relationship in the feature of responsive to receiving a request at the server for attributes for an entity from a client, generating a method call for the utility method, wherein the method call identifies the entity and a response object name. Instead, paragraph 39 of *Walker* discloses the getFieldDescriptions method, which is used to define parameters within a JAVA class to facilitate conversion between the JAVA class and XML. However, *Walker* fails to disclose a temporal aspect as to when a getFieldDescriptions method should be used to define how a JAVA class should be converted. *Walker* discloses only the getFieldDescriptions method must be used at some point before the JAVA/XML conversion so that the defined parameters can facilitate the conversion.

On the other hand, claim 1 recites a "responsive to" relationship in the feature of responsive to receiving a request at the server for attributes for an entity from a client, generating a method call for the utility method, wherein the method call identifies the entity and a response object name. Claim 1 links generating a method call for the utility method with receiving a request at the server for attributes for an entity from a client. As shown in Section II.A. above, *Walker* fails to disclose the feature of receiving a request at the server for attributes for an entity from a client. Because *Walker* does not disclose the condition to which the generating step is responsive, *Walker* does not disclose any relationship, let alone a "responsive to" relationship, between the generating step and receiving a request at the server for attributes for an entity from a client, as in claim 1.

Additionally, even assuming, *arguendo*, that *Walker* does disclose both receiving a request at the server for attributes for an entity from a client and the feature of generating a method call for the utility method, *Walker* still fails to disclose a "responsive to" relationship between the generating step and a received request, as claimed. For example, neither the cited portion nor any other portion of *Walker* discloses that using the getFieldDescriptions method to define JAVA class parameters is related to a request, let alone responsive to a request. Instead, *Walker* implies that parameters must be defined at some point before the conversion process begins. For example, *Walker* discloses that "[t]he FieldDescription class provides a set of information needed by the API to convert between JAVA and XML representations," but fails to disclose that the getFieldDescriptions method associated with the FieldDescription class defines this information in response to any request. *Walker*, paragraph 43. Because *Walker* fails to disclose that the generating step in the claimed feature is responsive to any request, *Walker* fails to disclose a "responsive to" relationship in the feature of responsive to receiving a request at the server for attributes for an entity from a client, generating a method call for the utility method, wherein the method call identifies the entity and a response object name. Accordingly, *Walker* does not disclose all the features of claim 1.

**II.D. Conclusion as to Anticipation**

Because claim 1 is representative of claim 13 and amended claim 22, the same distinctions between claim 1 and the reference also applies to claim 13 and amended claim 22. Additionally, because claim 2-6, 8-12, 14-21 depend from claims 1 and 13, at least the same distinctions between *Walker* and claims 1 and 13 apply for these claims as well. Additionally, claims 2-6, 8-12, 14-21 claim other additional combinations of features not disclosed by the reference. Therefore, the rejection of claims 1-6 and 8-22 under 35 U.S.C. § 102 has been overcome.

Furthermore, *Walker* does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. *Walker* actually teaches away from the presently claimed invention because it teaches a mechanism for converting between JAVA classes and XML without any disclosed need for a client or a server, as opposed to receiving a request at the server for attributes for an entity from a client, or returning the response object to a client, as in the presently claimed invention. Absent the Examiner pointing out some teaching or incentive to implement *Walker* and the teaching of a client and a server, as claimed, one of ordinary skill in the art would not be led to modify *Walker* to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify *Walker* in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

## III. 35 U.S.C. § 102, Anticipation; Claim 7

The Examiner rejected claim 7 under 35 U.S.C. § 102 as anticipated by *Walker*. The rejection is respectfully traversed.

Claim 7 is as follows:

> 7. The method of claim 6, wherein the at least one data item is retrieved from the database through a structured query language interface.

With regard to claim 7, the Examiner states that:

> Per claim 7, Walker discloses the method of claim 6, wherein the at least one data item is retrieved from the' database through a structured query language interface (e.g. FIG. 5, element 535 and related text and para [0051] ". . . GetMethod parameter is used . . . retrieve this field...").

Because amended claim 7 depends from claim 1, the same distinctions between *Walker* and the claimed invention in claim 1 can also be made for amended claim 7. Additionally, amended claim 7 claims other additional combinations of features not disclosed by *Walker*.

*Walker* fails to disclose the feature wherein the at least one data item is retrieved from the database through a structured query language interface. The Examiner asserts otherwise, citing various portions of *Walker*. Each of these portions will be addressed in turn. Applicant's first address the following portion of *Walker*:

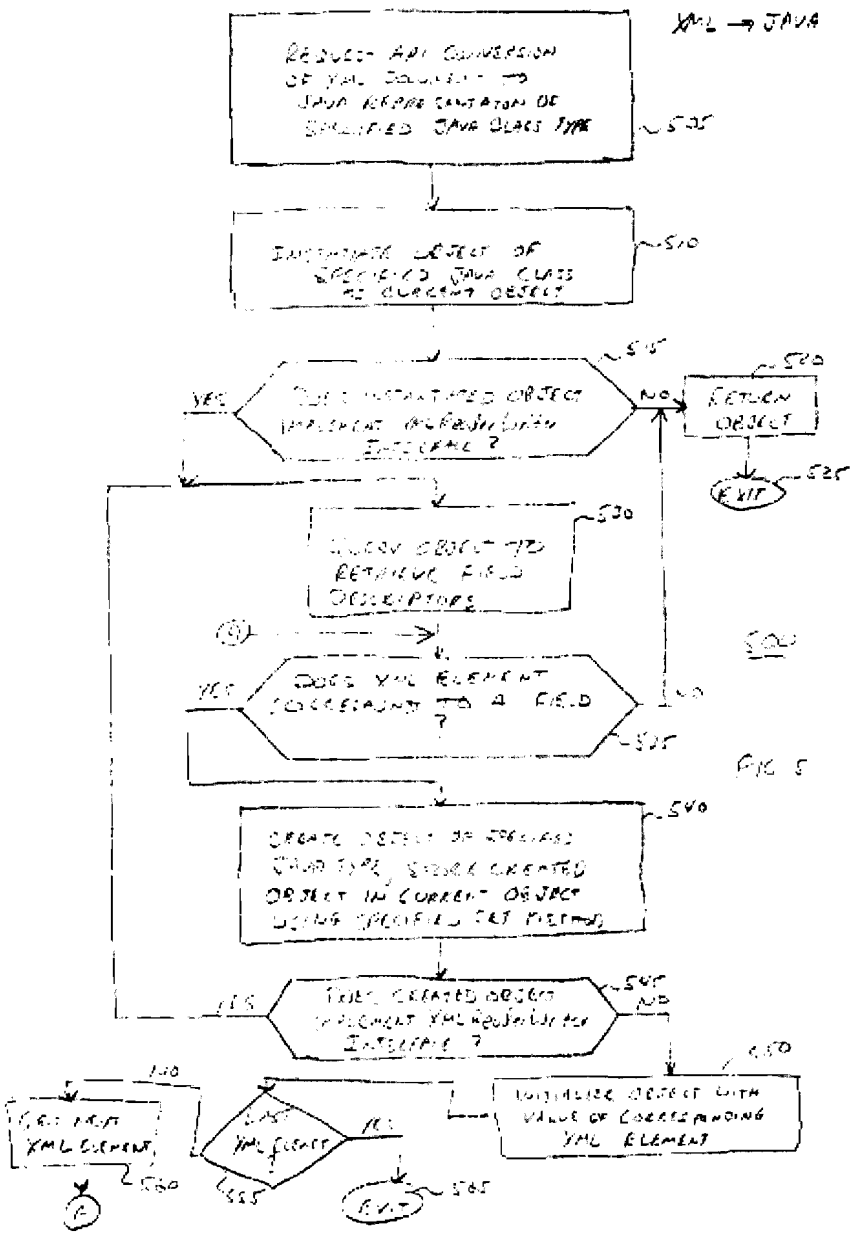



FIG 5

*Walker*, figure 5.

> [0115] At step 530, the instantiated object is queried to retrieve field descriptors. At step 535, a query is made as to whether a first (or next) XML element to be processed corresponds to one of the retrieved field descriptors. If the query at step 535 is answered negatively, then the method 500 proceeds to step 520, where the instantiated object is retrieved, and to step 525 where the method 500 is exited. If the query at step 535 is answered affirmatively, then the method 500 proceeds to step 540.

*Walker*, paragraph 115.

Neither the cited portions nor any other portion of *Walker* discloses the feature wherein the at least one data item is retrieved from the database through a structured query language interface. The cited

portions of *Walker* disclose steps in *Walker's* invention that convert XML into JAVA representations. Step 535, which the Examiner cites, discloses a query as to whether an XML element to be processed corresponds to one of the retrieved field descriptors. However, the cited portions fail to disclose any structured query language (SQL) interface.

On the other hand, claim 7 recites the feature wherein the at least one data item is retrieved from the database through a structured query language interface. SQL pertains to a well-known database management programming language. *Walker* fails to disclose using the SQL language for any reason, let alone to retrieve at least one data item from a database, as claimed. *Walker's* failure to disclose SQL is expected because *Walker's* invention does not pertain to the management of databases, for which SQL was created. Because *Walker* fails to disclose the use of SQL, *Walker* also fails to disclose any SQL interface, as claimed. Therefore, *Walker* fails to disclose the feature wherein the at least one data item is retrieved from the database through a structured query language interface.

Additionally, the Examiner cites the following portion of *Walker*:

> [0051] The TagName parameter is used to identify an XML element tag for a corresponding JAVA field being converted to XML. The ObjectClass parameter is used to specify the JAVA class to be instantiated when constructing the field from an XML document or representation. The GetMethod parameter is used to identify the JAVA method invoked to retrieve this field. The SetMethod parameter is used to specify the JAVA method invoked to retrieve a described method.

*Walker*, paragraph 51.

However, the cited portion of *Walker*, like all other portions of *Walker*, fails to disclose SQL. The cited portion discloses only using parameters to define elements in a JAVA class. These parameters are not part of SQL. *Walker's* failure to disclose SQL precludes any disclosed need for an SQL interface, as claimed. Also, for the same reason stated above, *Walker* has no disclosed need to disclose an SQL interface because *Walker* does not pertain to database management. Therefore, *Walker* fails to disclose the feature wherein the at least one data item is retrieved from the database through a structured query language interface. Accordingly, the rejection of amended claims 7 under 35 U.S.C. § 102 has been overcome.

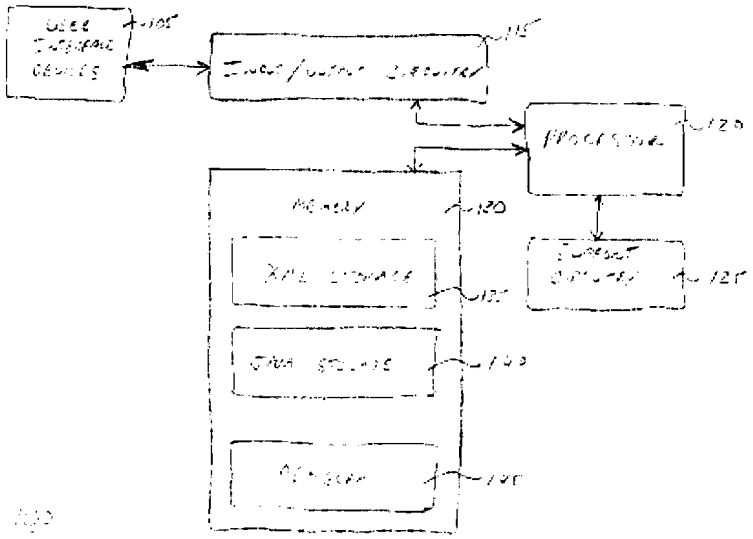
## IV. 35 U.S.C. § 102, Anticipation; New Claim 23

New claim 23 has been added. *Walker* does not anticipate claim 23 because *Walker* does not disclose each and every feature of claim 23. Claim 23 is as follows:

> 23. The method of claim 1, wherein the server is located at a first computer system, and wherein the client is located at a second computer system, and wherein the first computer system is separate from the second computer system.

Because claim 23 depends from claim 1, the same distinctions between *Walker* and the claimed invention in claim 1 can also be made for claim 23. Additionally, claim 23 claims other additional combinations of features not disclosed by *Walker*.

*Walker* fails to disclose the feature wherein the server is located at a first computer system, and wherein the client is located at a second computer system, and wherein the first computer system is separate from the second computer system. Instead, *Walker* discloses a JAVA/XML conversion mechanism, embodied as a computer program, which executes on a single computer. For example, *Walker* discloses that "[i]t will be appreciated by those skilled in the art that one embodiment is implemented as a program product for use with a computer system such as, for example, the system 100 shown in FIG. 1." Figure 1 is reproduced to the left for the Examiner's convenience. *Walker* consistently discloses implementing his invention only on computer system 100, and fails to disclose otherwise.



On the other hand, claim 7 recites the feature wherein the server is located at a first computer system, and wherein the client is located at a second computer system, and wherein the first computer system is separate from the second computer system. Because *Walker* fails to disclose implementing his invention on more than one computer system, *Walker* does not disclose a client and a server located on two separate computer systems. *Walker* focuses on the implementation of a JAVA/XML conversion mechanism that executes on a single computer system, and fails to disclose or address implementing his invention on more than one computer system. Also, as shown in Section II, *Walker* fails to disclose the features of a client and a server as claimed in claim 1. Because *Walker* fails to disclose the features of a client and a server, as claimed in claim 1, *Walker* cannot disclose the features of a client and a server located on two separate computer systems either. Therefore, *Walker* fails to disclose the feature wherein the server is located at a first computer system, and wherein the client is located at a second computer system, and wherein the first computer system is separate from the second computer system. Accordingly, no anticipation rejection can be made against claim 23.

## V. Conclusion

The subject application is patentable over the cited reference and should now be in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: February 12, 2007

Respectfully submitted,

/Theodore D. Fay III/

Theodore D. Fay III
Reg. No. 48,504
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicant

TF/KA